T13 - LENGUAJES DE INTERROGACIÓN DE BASES DE DATOS.

1. LENGUAJES DE INTERROGACION DE BASES DE DATOS 2

1.1 EL ÁLGEBRA RELACIONAL 2
1.1.1 OPERACIONES FUNDAMENTALES 2
1.1.1.1 LA OPERACIÓN SELECCIÓN 2
1.1.1.2 LA OPERACIÓN PROYECCIÓN 3
1.1.1.3 COMPOSICIÓN DE OPERACIONES RELACIONALES 3
1.1.1.4 LA OPERACIÓN UNIÓN 3
1.1.1.5 LA OPERACIÓN DIFERENCIA DE CONJUNTOS 4
1.1.1.6 LA OPERACIÓN PRODUCTO CARTESIANO 4
1.1.1.7 LA OPERACIÓN RENOMBRAMIENTO 4
1.1.2 OTRAS OPERACIONES DERIVADAS 5
1.1.2.1 LA OPERACIÓN INTERSECCIÓN DE CONJUNTOS 5
1.1.2.2 LA OPERACIÓN UNIÓN NATURAL 5
1.1.2.3 LA OPERACIÓN ASIGNACIÓN 6
1.2 EL CÁLCULO RELACIONAL DE TUPLAS 6
1.3 EL CÁLCULO RELACIONAL DE DOMINIOS 7

2. EL LENGUAJE SQL (STRUCTURED QUERY LANGUAGE) 7

2.1 LENGUAJE DE DEFINICIÓN DE DATOS (DDL) 8
2.1.1 TIPOS BÁSICOS DE DATOS 8
2.1.2 CREACIÓN Y BORRADO DE BASES DE DATOS 9
2.1.3 CREACIÓN, MODIFICACIÓN Y BORRADO DE TABLAS 9
2.1.4 DEFINICIÓN DE VISTAS 9
2.1.5 CREACIÓN Y BORRADO DE ÍNDICES 10
2.1.6 DEFINICIÓN DE CLAVES REFERENCIALES 10
2.2 LENGUAJE DE MANIPULACIÓN DE DATOS (DML) 11
2.2.1 INSERCIÓN, ACTUALIZACIÓN Y BORRADO DE FILAS 11
2.2.2 OPERACIONES DE CONSULTA 11
2.3 LENGUAJE DE CONTROL DE DATOS (DCL) 13
2.3.1 CONTROL DE ACCESO A LOS DATOS 13
2.3.2 CONTROL DE INTEGRIDAD 14

3. ESTANDARES DE CONECTIVIDAD: ODBC Y JDBC 14

3.1 ODBC 14
3.2 JDBC 16

4. CONCLUSIÓN 17

5. BIBLIOGRAFÍA 18

6. ESQUEMA – RESUMEN 19

1. LENGUAJES DE INTERROGACION DE BASES DE DATOS

Un lenguaje de interrogación o consulta es un lenguaje en el que un usuario solicita información de la base de datos. Estos lenguajes suelen ser de un nivel superior que el de los lenguajes de programación habituales. Los lenguajes de consulta pueden clasificarse, tal como vimos al estudiar los LMD, en dos grupos:

- **Lenguajes procedurales o procedimentales:** El usuario instruye al sistema para que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado.
- **Lenguajes no procedurales o no procedimentales:** El usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.

En este epígrafe vamos a estudiar los lenguajes formales de consulta o lenguajes "puros". Los tres que se estudian no son cómodos de usar, pero a cambio sirven como base formal para lenguajes de consulta que si resultan cómodos.

El primer lenguaje de consulta, el álgebra relacional, se estudia en detalle. El álgebra relacional forma la base del lenguaje de consulta SQL, ampliamente usado y que también estudiaremos en el epígrafe siguiente. A continuación se proporciona una visión general de otros dos lenguajes formales: el cálculo relacional de tuplas y el cálculo relacional de dominios, que son lenguajes declarativos de consulta basados en la lógica matemática. El cálculo relacional de dominios es la base del lenguaje de consulta QBE.

1.1 EL ÁLGEBRA RELACIONAL

El álgebra relacional es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

Como ya vimos al estudiar el modelo relacional, las operaciones asociadas a este modelo de datos forman el álgebra relacional. Se puede demostrar matemáticamente que ésta álgebra es completa, o sea, que por medio de ella se puede realizar cualquier acceso a la base de datos.

Las operaciones fundamentales del álgebra relacional son: *selección, proyección, unión, diferencia de conjuntos, producto cartesiano y renombramiento.* Además de estas operaciones fundamentales hay otras operaciones que se definen a partir de las fundamentales, tales como: *intersección de conjuntos, unión natural y asignación.*

Veamos ahora con más detalle cada una de estas operaciones. Los ejemplos en los que nos vamos a apoyar hacen referencia a las tablas ejemplo que aparecen en el apartado 3.2.1: Alumno, Asignatura y Cursa.

1.1.1 OPERACIONES FUNDAMENTALES

Se dividen estas operaciones en dos tipos:

- **Unarias:** Porque operan con una sola relación o tabla. Son: selección, proyección y renombramiento.
- **Binarias:** Porque operan con dos relaciones. Son: unión, diferencia de conjuntos y producto cartesiano.

1.1.1.1 LA OPERACIÓN SELECCIÓN

La operación **selección** selecciona tuplas que satisfacen un predicado dado. Se utiliza la letra griega sigma minúscula (σ) para denotar la selección. El predicado aparece como subíndice de σ. La relación de entrada es el argumento que aparece entre paréntesis a continuación de σ.

Por lo tanto, la definición formal dice: Si P es un predicado compuesto por operadores lógicos, aritméticos y de comparación y sus operandos son los valores de los atributos de una relación R, entonces la selección $\sigma_P(R)$ es el conjunto de tuplas de la relación R que hacen verdadera la condición establecida por el predicado P.

Por ejemplo para seleccionar las tuplas de la relación Alumno en que el nombre es “Ana”, hay que escribir:

$$\sigma_{Nom=”Ana”} (Alumno)$$

En general, se permite las comparaciones que utilizan los signos: =, ≠, <, ≤, > o ≥, en el predicado de selección. Además se pueden combinar varios predicados en uno mayor utilizando las conectivas *y(^)* y *o(v)*. El predicado de selección puede incluir comparación entre dos atributos.

1.1.1.2 LA OPERACIÓN PROYECCIÓN

La operación proyección es una operación unaria que dada una relación de entrada, devuelve todos atributos de dicha relación que aparecen en los argumentos de dicha operación. Dado que las relaciones son conjuntos, se eliminan todas las filas duplicadas.

La proyección se denota por la letra griega pi (π). Se crea una lista de atributos que se desea que aparezcan en el resultado, como subíndice de π. La relación de entrada es el argumento que aparece entre paréntesis a continuación de π.

Por lo tanto, la definición formal dice: Si x es un subconjunto de atributos de la relación R, entonces la proyección $\pi_x(R)$ es la relación formada por las columnas de R correspondientes a los atributos x.

Por ejemplo, la consulta para crear una lista de todos los DNI y números de matrícula de la relación Alumno, se escribe:

$$\pi_{DNI,NMat} (Alumno)$$

1.1.1.3 COMPOSICIÓN DE OPERACIONES RELACIONALES

Es importante el hecho de que el resultado de una operación relacional sea también una relación.

En general, dado que el resultado de una operación de álgebra relacional es del mismo tipo (relación) que los datos de entrada, las operaciones del álgebra relacional pueden componerse para formar una **expresión del álgebra relacional.** La composición de operaciones del álgebra relacional para formar expresiones del álgebra relacional es igual que la composición de operaciones aritméticas (como +, -, * y ÷) para formar expresiones aritméticas.

Por ejemplo, la consulta para encontrar el DNI y número de matrícula de los alumnos en que el nombre es “Ana”, es una expresión del álgebra relacional, que se escribe:

$$\pi_{DNI,NMat} (\sigma_{Nom=”Ana”} (Alumno))$$

1.1.1.4 LA OPERACIÓN UNIÓN

Consideramos una consulta para averiguar el DNI de todos los alumnos que están en la tabla Alumnos, en la tabla Cursa o en ambas. Se conoce la manera de obtener los DNI de cada una de las tablas:

$$\pi \text{ DNI (Alumno)} \text{ y } \pi \text{ DNI (Cursa)}$$

Sin embargo, para contestar a la consulta hace falta la **unión** de estos dos conjuntos, es decir, hacen falta todos los DNI que aparezcan en algunas de las dos relaciones o en ambas. Estos datos se pueden averiguar mediante la operación binaria **unión**, denotada, como en la teoría de conjuntos, por U. En el resultado, dado que las relaciones son conjuntos, se eliminan los valores duplicados. Por lo tanto, la expresión buscada es:

$$\pi_{DNI}(Alumno) \cup \pi_{DNI}(Cursa)$$

En general, se debe asegurar que las uniones se realicen entre relaciones *compatibles,* es decir, que deben cumplir las dos condiciones siguientes:

- Las dos relaciones deben ser de la misma *aridad*, es decir, deben tener el mismo número de atributos.
- Los dominios de los atributos, deben ser iguales.

Por lo tanto, la definición formal dice: La unión de dos relaciones R y S (R U S) es el conjunto formado por todas las tuplas de R más todas las tuplas de S. Este operador sólo se puede aplicar a relaciones del mismo grado y con los mismos atributos.

1.1.1.5 LA OPERACIÓN DIFERENCIA DE CONJUNTOS

La operación diferencia de conjuntos, denotada por -, permite buscar la tuplas que estén en una relación pero no en otra. La definición formal dice: La **diferencia** de dos relaciones R y S (R - S) es el conjunto formado por todas las tuplas de R que no están en S.

Este operador, al igual que el operador unión, solo puede realizarse entre relaciones *compatibles*. Por lo tanto el operador diferencia sólo se puede aplicar a relaciones del mismo grado y con los mismos atributos.

Por ejemplo, la consulta para encontrar el DNI de los alumnos que están en la tabla Alumno, pero no están en la tabla Cursa, al no cursar ninguna asignatura, se escribe:

$$\pi_{DNI}(Alumno) - \pi_{DNI}(Cursa)$$

En nuestro ejemplo, el resultado sería la relación vacía.

1.1.1.6 LA OPERACIÓN PRODUCTO CARTESIANO

La operación producto cartesiano denotada, por un aspa (x), permite combinar información de cualesquiera dos relaciones. Hay que considerar dos posibles problemas:

- Si las dos relaciones de entrada tienen un atributo con el mismo nombre, se adjunta a dicho atributo el nombre de la relación, para así distinguir uno de otro.
- Si el nombre de las dos relaciones de entrada es el mismo (producto cartesiano de una relación consigo misma) o si se utiliza el resultado de una expresión del álgebra relacional en un producto cartesiano, se debe dar un nuevo nombre a una de las relaciones o a la expresión del álgebra relaciona utilizando una operación de renombramiento que veremos en el apartado siguiente.

La definición formal dice: El **producto cartesiano** de dos relaciones R y S, de grados *m* y *n* respectivamente, se denota R x S y es el conjunto formado por todas las posibles tuplas de *m* + *n* atributos en las que los *m* primeros atributos son de R y los *n* restantes pertenecen a S.

1.1.1.7 LA OPERACIÓN RENOMBRAMIENTO

A diferencia de las relaciones de base de datos, los resultados de las expresiones del álgebra relacional no tienen un nombre que se pueda utilizar para referirse a ellas. Resulta, por lo tanto, útil ponerles nombre. La operación **renombramiento** denotado por la letra griega rho (ρ), permite realizar esta tarea.

La definición formal dice: Dada una expresión E del álgebra relacional, la expresión $\rho_x(E)$, devuelve el resultado de la expresión E con nombre x.

Las relaciones por si mismas, también se consideran expresiones triviales del álgebra relacional. Por lo tanto, también se puede aplicar la operación renombramiento a una relación dada, para obtener la misma relación con un nombre nuevo.

Por ejemplo, vamos a dar el nombre Ana a la expresión $\pi_{DNI,NMat}\ (\sigma_{Nom="Ana"}\ (Alumno))$, se escribe:

$$\rho_{Ana}\ (\pi_{DNI,NMat}\ (\sigma_{Nom="Ana"}\ (Alumno)))$$

1.1.2 OTRAS OPERACIONES DERIVADAS

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta del álgebra relacional. Sin embargo si uno se limita únicamente a las operaciones fundamentales, algunas consultas habituales, resultan ser algo más complejas de expresar. Por este motivo, se definen otras operaciones que no añaden potencia al álgebra, pero que simplifican las consultas habituales.

A partir de las operaciones básicas, es posible definir otras operaciones derivadas tales como la intersección, la unión natural, y la asignación.

1.1.2.1 LA OPERACIÓN INTERSECCIÓN DE CONJUNTOS

La operación **intersección** de conjuntos denotada por ∩, permite buscar la tuplas que estén al tiempo en las dos relaciones sobre las que actúa.

Se observa que esta operación no es fundamental y no añade potencia al álgebra relacional, ya que puede ser expresada en función de la operación diferencia de conjuntos, de la manera siguiente:

$$R \cap S = R - (R - S)$$

Por ejemplo, la consulta para encontrar los DNI de los alumnos que están en la tabla Alumno y que están en la tabla Cursa, se escribe:

$$\pi_{DNI}\ (Alumno) \cap \pi_{DNI}\ (Cursa)$$

1.1.2.2 LA OPERACIÓN UNIÓN NATURAL

Suele resultar deseable simplificar ciertas consultas que exigen producto cartesiano. Generalmente, las consultas que implican producto cartesiano incluyen un operador selección sobre el resultado del producto cartesiano.

La **unión natural** es una operación binaria que permite combinar ciertas selecciones y un producto cartesiano en una sola operación. Se denota por el símbolo de la "reunión" $\bowtie$. La operación unión natural forma un producto cartesiano de sus dos argumentos, realiza una selección forzando la igualdad de los atributos que aparecen en ambas relaciones y, finalmente elimina los atributos duplicados.

1.1.2.3 LA OPERACIÓN ASIGNACIÓN

En ocasiones resulta conveniente escribir una expresión de álgebra relacional por partes utilizando la asignación a una variable de relación temporal.

La operación **asignación**, denotada por ←, actúa de manera parecida a la asignación de los lenguajes de programación.

Por ejemplo, se puede realizar una asignación de la expresión $\pi_{DNI,NMat}(\sigma_{Nom="Ana"}(Alumno))$ a una variable de relación temporal, llamada temp1:

$$temp1 \leftarrow \pi_{DNI,NMat}(\sigma_{Nom="Ana"}(Alumno))$$

Con la operación asignación se puede escribir las consultas como programas secuenciales consistentes en una serie de asignaciones seguida de una expresión cuyo valor se muestra como resultado de la consulta.

1.2 EL CÁLCULO RELACIONAL DE TUPLAS

Cuando escribimos una expresión de álgebra relacional proporcionamos una serie de procedimientos que generan la respuesta a la consulta. El cálculo relacional de tuplas, en cambio, es un lenguaje de consulta **no precedimental**. Describe la información deseada sin dar un procedimiento específico para obtenerla.

Las consultas se expresan en el cálculo relacional de tuplas como: $\{t \mid P(t)\}$, es decir, son el conjunto de todas las tuplas tales que el predicado P es cierto para *t*. Se utiliza la notación *t*[A] para denotar el valor de la tupla *t* en el atributo A y $t \in R$ para denotar que la tupla *t* está en la relación R.

Para poder hacer una definición formal del cálculo relacional de tuplas, debemos conocer los tres conceptos siguientes:

- Constructor “existe” de la lógica matemática. La notación $\exists\ t \in R(Q(t))$ significa: existe una tupla *t* en la relación R tal que el predicado Q(*t*) es verdadero.
- Implicación denotada por $\Rightarrow$, es decir P implica Q, se escribe: $P \Rightarrow Q$
- Constructor “para todo” de la lógica matemática. La notación $\forall\ t \in R(Q(t))$ significa: Q es verdadera para todas las tuplas *t* de la relación R.

Ahora podemos dar una definición formal del cálculo relacional de tuplas.

Como ya hemos dicho, las expresiones del cálculo relacional de tuplas son de la forma: $\{t \mid P(t)\}$ donde P es un predicado o una fórmula. En una formula pueden aparecer varias variables tupla. Se dice que una **variable tupla** es una variable libre a menos que esté cuantificada mediante $\exists$ o $\forall$.

Las fórmulas del cálculo relacional de tuplas se construyen con **átomos**. Los átomos tienen una de las formas siguientes:

- $s \in R$, donde *s* es una variable tupla y R es una relación.
- $s[x]\ \theta\ u[y]$, donde *s* y *u* son variables tuplas, x es un atributo en el que está definida *s,* y es un atributo en el está definida *u* y θ es un operador de comparación (<, <=, >, >=, <>, =). Es necesario que los atributos x e y tengan dominios cuyos miembros puedan compararse mediante θ.
- $s[x]\ \theta\ c$, donde *s* es una variable tupla, x es un atributo en el que está definida *s*, θ es un operador de comparación y *c* es una constante en el dominio de atributo x.

Las fórmulas se construyen a partir de los átomos utilizando las reglas siguientes:

- Si P_1 es una fórmula, también lo son $\neg P_1$ y (P_1).
- Si P_1 y P_2 son fórmulas, también lo son $P_1 \vee P_2$, $P_1 \wedge P_2$ y $P_1 \Rightarrow P_2$.
- Si $P_1(s)$ es una fórmula que contiene una variable tupla libre *s*, y R es una relación:

  $\exists\ s \in R(P_1(s))$ y $\forall\ s \in R(P_1(s))$ también son fórmulas.

1.3 EL CÁLCULO RELACIONAL DE DOMINIOS

Hay una segunda forma de cálculo relacional denominada cálculo relacional de dominios. Esta forma utiliza variables de dominio que toman sus valores del dominio de un atributo, en vez de tomarlos de una tupla completa. El cálculo relacional de dominios, sin embargo, se halla estrechamente relacionado con el cálculo relacional de tuplas.

Las expresiones del cálculo relacional de dominios son de la forma:
$\{< x_1, x_2,\ldots, x_n > \mid P(x_1, x_2,\ldots, x_n)\}$, donde $x_1, x_2,\ldots, x_n$ representan las variables de dominio, P representa una fórmula compuesta de átomos, como era el caso en el cálculo relacional de tuplas.
Los átomos del cálculo relacional de dominios tienen una de las formas siguientes:

- $< x_1, x_2,\ldots, x_n > \in R$, donde R es una relación con *n* atributos y $x_1, x_2,\ldots, x_n$ son variables de dominio o constantes de dominio.
- $x\ \theta\ y$, donde x e y son variables de dominio y θ es un operador de comparación (<, <=, >, >=, <>, =). Se exige que los atributos x e y tengan dominios que puedan compararse mediante θ.
- $x\ \theta\ c$, donde x es una variable de dominio, θ es un operador de comparación y *c* es una constante de dominio del atributo para el que x es una variable de dominio.

Las fórmulas se construyen a partir de los átomos utilizando las reglas siguientes:

- Un átomo es una fórmula.
- Si P_1 es una fórmula, también lo son $\neg P_1$ y (P_1).
- Si P_1 y P_2 son fórmulas, también lo son $P_1 \vee P_2$, $P_1 \wedge P_2$ y $P_1 \Rightarrow P_2$.
- Si $P_1(x)$ es una fórmula en x, donde x es una variable de dominio:

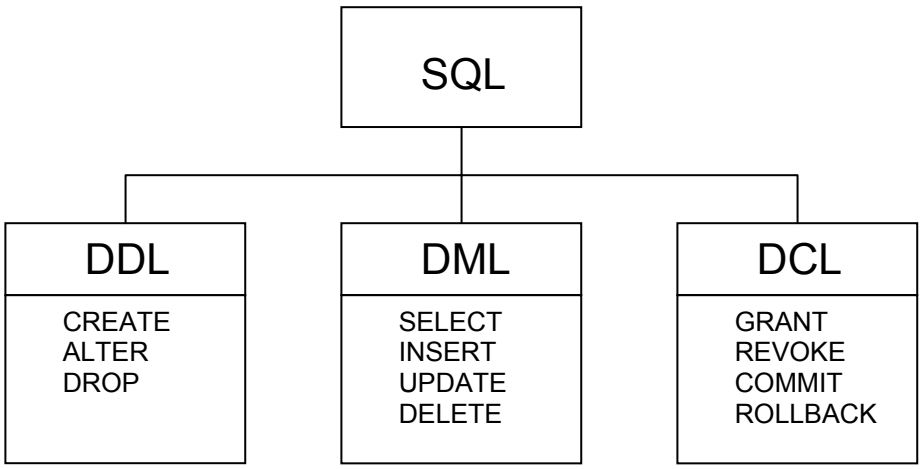
  $\exists\ x\ (P_1(x))$ y $\forall\ x\ (P_1(x))$ también son fórmulas.

2. EL LENGUAJE SQL (Structured Query Language)

Los lenguajes formales descritos en el epígrafe anterior proporcionan una notación concisa para la representación de las consultas. Sin embargo, los sistemas de bases de datos comerciales necesitan un lenguaje de consulta cómodo para el usuario. Vamos a estudiar el lenguaje comercial que actualmente tiene mayor influencia, SQL. SQL es una combinación de álgebra relacional y construcciones de cálculo relacional.

Aunque el lenguaje SQL se considere un lenguaje de consultas, contiene muchas otras capacidades además de la consulta en base de datos. Incluye características para definir la estructura de los datos, para la modificación de los datos en la base de datos y para especificación de restricciones de integridad.

El lenguaje SQL es un lenguaje de alto nivel para dialogar con los SGBD-R. Como todo lenguaje de un SGBD, esta formado por tres componentes claramente diferenciados, según muestra la figura:



Destacamos algunas de las características principales del lenguaje SQL:

- Utilizado por todo tipo de usuarios:
  - Administradores de BDR.
  - Programadores.
  - Usuarios Finales.
- Lenguaje no procedimental: Se especifica QUÉ se quiere obtener, sin decir CÓMO.
- Permite especificar cualquier consulta.

No se pretende proporcionar un manual de usuario completo para SQL. Por el contrario, se van a presentar las construcciones y conceptos fundamentales de SQL. Las distintas implementaciones de SQL pueden diferenciarse en detalles, o pueden admitir sólo un subconjunto del lenguaje completo, según el SGBDR que se utilice.

2.1 LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

2.1.1 TIPOS BÁSICOS DE DATOS

- Datos Alfanuméricos o Cadenas de Caracteres:
  - CHAR(longitud), donde: longitud = número máximo de caracteres del campo
  - VARCHAR(longitud)
- Datos Numéricos:
  - SMALLINT , INTEGER
  - DECIMAL ó DECIMAL(precisión, decimal), donde: precisión = número de dígitos del número y decimal = número de dígitos decimales del nº decimal
  - REAL
  - FLOAT
- Datos tipo fecha y tiempo:
  - DATE: Se puede elegir entre varios formatos.
  - TIME: También tiene diferentes formatos.

- TIMESTAMP: Su valor es: fecha + tiempo + nanosegundos.

2.1.2 CREACIÓN Y BORRADO DE BASES DE DATOS

- Creación de una Base de Datos: CREATE DATABASE nombre_base_datos;
- Borrado de la Base de Datos: DROP DATABASE nombre_base_datos;

2.1.3 CREACIÓN, MODIFICACIÓN Y BORRADO DE TABLAS

- Creación de tablas (formato básico):

CREATE TABLE nombre_tabla
(<definición_atributo_1> [NOT NULL] [CHECK Condicion],
<definición_atributo_2> [NOT NULL] [CHECK Condicion],
.....
<definición_atributo_n> [NOT NULL] [CHECK Condicion],
[PRIMARY KEY (ListadeAtributos)]); donde:

  - definición_atributo = nombre_atributo tipo_dato (tamaño)
  - NOT NULL: no se permiten valores nulos en la columna
  - ListadeAtributos: uno o más atributos separados por comas

- Modificación de tablas:
  - Añadir un nuevo atributo:

ALTER TABLE <nombre_tabla>
ADD <def_atributo>|<def_integridad>;

  - Modificar un atributo ya existente:

ALTER TABLE <nombre_tabla>
ALTER <Atributo> TYPE <Nuevo_tipo>;

  - Borrar un atributo ya existente:

ALTER TABLE <nombre_tabla>
DROP <Atributo>;

- Eliminación de tablas:

DROP TABLE <nombre_tabla>;

2.1.4 DEFINICIÓN DE VISTAS

Una vista es una estructura tabular no física (tabla virtual), que permite consultar y/o modificar datos de la tabla real.

Las principales características de las vistas son:

- Se utilizan como si fuesen tablas reales.
- No contienen datos propios.
- No tienen asociada estructura física.

Las ventajas del uso de vistas son:

- Menor complejidad en consultas: Permiten obtener igual información de forma más simple.
- Aumento confidencialidad: Permiten acceder sólo a ciertos datos de las tablas reales.

Las vistas se pueden crear y borrar con las siguientes sentencias:

- Creación de vistas:

```
CREATE VIEW <nombre_vista> [(<lista_atributos>)]
AS ( <cláusula SELECT> );
```

Las filas de la vista serán aquellas que resulten de ejecutar la consulta sobre la que está definida.

- Eliminación de vistas:

```
DROP VIEW <nombre_vista>;
```

2.1.5 CREACIÓN Y BORRADO DE ÍNDICES

Es el sistema el encargado de utilizar los índices, para optimizar el acceso a los datos. el usuario sólo puede crear o eliminar índices, pero no indicar su utilización.

- Creación de índices:

```
CREATE [UNIQUE] INDEX <nombre_índice>
ON <nombre_tabla> (<lista_atributos>);
```

- Eliminación de índices:

```
DROP INDEX <nombre_índice>;
```

2.1.6 DEFINICIÓN DE CLAVES REFERENCIALES

- En la creación de la tabla (formato completo):

```
CREATE TABLE nombre_tabla
( nombre_columna tipo_columna [NOT NULL] [DEFAULT Valor]
[CHECK Condicion ],
..... ,
nombre_columna tipo_columna [NOT NULL] [DEFAULT Valor]
[CHECK Condicion ],
[PRIMARY KEY (lista_de_columnas)] ,
[FOREIGN KEY (lista_de_columnas)
REFERENCES nombre_de_tabla (lista_de_columnas)
ON UPDATE [NO ACTION | SET DEFAULT | SET NULL |
CASCADE]
ON DELETE [NO ACTION | SET DEFAULT | SET NULL |
CASCADE]
FOREIGN KEY ..... );
```

2.2 LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

2.2.1 INSERCIÓN, ACTUALIZACIÓN Y BORRADO DE FILAS

- Inserción de filas:
  - Inserción de una fila:

INSERT INTO <nombre_tabla> [(<lista_de _atributos>)]
VALUES (<valor_1>, <valor_2>,...,<valor_n>);

  - Inserción de varias filas:

INSERT INTO <nombre_tabla> [(<lista_de_atributos>)]
(<cláusula SELECT>); La cláusula "SELECT" especifica una consulta cuyo resultado (filas) se insertará en la tabla especificada.

- Modificación de filas:

UPDATE <nombre_tabla>
SET <atributo_1> = <valor_1>,
<atributo_2> = <valor_2>,
.....
<atributo_n> = <valor_n>
[WHERE <condición>]; La modificación afectará a todas las filas que cumplan la condición, si se especifica ésta. Si no se especifica condición, la modificación afectará a todas las filas de la tabla.

- Eliminación de filas:

DELETE
FROM <nombre_tabla>
[WHERE <condición>]; No se pueden eliminar partes de una fila. Si no aparece la cláusula "WHERE" se eliminarán todas las filas de la tabla, no eliminándose la definición de ésta en el esquema.

2.2.2 OPERACIONES DE CONSULTA

- Sintaxis de la sentencia:

SELECT [DISTINCT] <expresión>
FROM <lista_de_tablas>
[WHERE <condición>]
[GROUP BY <lista_de_atributos>
[HAVING <condición_de_grupo>]]
[ORDER BY <lista_de_atributos> [ASC/DESC]]; donde:

  - SELECT: especifica la información que se desea obtener.
  - DISTINCT: elimina los valores repetidos.
  - FROM: indica las tablas o vistas en las que se encuentran los atributos implicados en la consulta.
  - WHERE: especifica la condición de búsqueda.
  - GROUP BY: permite agrupar el resultado.
  - HAVING: especifica una condición de grupo.
  - ORDER BY: permite ordenar el resultado.

- Operadores: Los operadores que se pueden utilizar para expresar condiciones de fila (cláusula WHERE) o de grupo (cláusula HAVING) son:
  - De comparación: <, <=, >, >=, <>, =

- Lógicos: AND, OR, NOT
- BETWEEN ... AND ...: establece una comparación dentro de un intervalo cerrado. También se puede utilizar NOT BETWEEN.
- LIKE: establece una comparación entre cadenas de caracteres, empleando los siguientes comodines: '%': sustituye a una cadena de caracteres cualquiera y '_': sustituye a un único carácter cualquiera. También se puede utilizar NOT LIKE.
- IN: comprueba la pertenencia de un valor a un conjunto dado.
- IS NULL: comprueba si un valor determinado es nulo (NULL). También se puede utilizar IS NOT NULL.
- Cuantificadores: ANY (alguno), ALL (todos)
- Existencial: EXISTS, Indica la existencia o no de un conjunto. También se puede utilizar NOT EXISTS.

□ Reglas de Evaluación de Operadores: El Orden de Evaluación es el siguiente:
- Operadores de Relación: BETWEEN, IN, LIKE, IS, NULL y después NOT, AND, OR.
- Se pueden utilizar paréntesis para establecer el orden de evaluación deseado por el usuario.

□ Consultas con UNION, DIFERENCIA e INTERSECCIÓN:
- Unión de conjuntos: operador UNION.
- Diferencia de conjuntos: operador MINUS.
- Intersección de conjuntos: operador INTERSECT.

□ Expresiones en la cláusula SELECT: No sólo se pueden seleccionar atributos, sino expresiones en las que aparezcan atributos y/o constantes y operadores aritméticos.

□ Funciones agregadas: Devuelven un valor único, numérico. No se pueden combinar, con columnas que devuelvan más de un valor, a menos que la consulta contenga una cláusula GROUP BY.
- COUNT (*): contador de tuplas (totalizador)
- COUNT (DISTINCT Atributo): contador de tuplas (parcial), no tiene en cuenta valores nulos ni duplicados.
- AVG(Atributo): media aritmética de un atributo numérico
- SUM(Atributo): suma de atributos o expresiones numéricas
- MAX(Atributo): valor máximo de un atributo o expresión numérica
- MIN(Atributo): valor mínimo de un atributo o expresión numérica

□ Cláusula GROUP BY: GROUP BY <lista_de_atributos>
- Agrupa el resultado, devolviendo una única fila por grupo.
- El agrupamiento no se realiza ordenado.
- Los atributos que aparezcan en GROUP BY, deben aparecer en la cláusula SELECT.

□ Cláusula HAVING: HAVING <condición_de_grupo>
- Siempre va acompañada de la cláusula GROUP BY.
- Especifica una condición de grupo.

□ Cláusula ORDER BY: ORDER BY <lista_de_atributos> [ASC | DESC]
- El resultado de la consulta se ordena en base a los atributos que se indiquen en la lista.
- Los atributos de ordenación deben aparecer en SELECT.

2.3 LENGUAJE DE CONTROL DE DATOS (DCL)

Este lenguaje se preocupa principalmente del control de acceso a los datos (seguridad) y del control de la integridad de los datos.

2.3.1 CONTROL DE ACCESO A LOS DATOS

- Niveles de acceso soportados por los SGBDR: Se establecen privilegios de acceso por los niveles siguientes:
  - Base de Datos.
  - Tabla.
  - Atributo.
  - Tupla.
- Concesión de Privilegios: Permite dar a los usuarios el acceso completo o restringido a la base de datos:

GRANT <privilegio_de_acceso>
[ON <lista_de_objetos>]
TO <lista_de_usuarios>
[WITH GRANT OPTION]; donde:

  - <privilegio_de_acceso>: CONNECT, RESOURCE, DBA, ALL PRIVILEGES, SELECT, UPDATE, INSERT, DELETE.
  - WITH GRANT OPTION concede permiso para que el usuario a su vez, conceda esos permisos a otros usuarios.
- Nivel de Base de Datos: El SGBDR chequea los privilegios del usuario al iniciar la sesión. Los posibles privilegios o permisos son:
  - CONNECT: Conectarse a la BDR.
  - RESOURCE: Crear objetos.
  - DBA: Ejecución de comandos restrictivos. Acceso a cualquier objeto. Privilegio RESOURCE implícito.
- Nivel de Tabla: Las tablas son propiedad del usuario que las creó. Los posibles privilegios o permisos son:
  - DELETE: Autoriza el borrado de tuplas.
  - INSERT: Autoriza la inserción de nuevas tuplas.
  - SELECT: Permite la realización de consultas.
  - UPDATE: Permite la actualización de tuplas.
  - ALL PRIVILEGES: Concede todos los privilegios.
- Niveles Atributo y Tupla: Se implantan a través de la definición de vistas.
  - Nivel de Atributo: Se crea una vista sin condiciones. Se establecen los permisos sobre la vista.
  - Nivel de Tupla: Se crea una vista con sólo las tuplas permitidas. Se establecen los permisos sobre la vista.
- Revocación de privilegios: Se utiliza para anular privilegios ya concedidos a los usuarios:

REVOKE <privilegio_de_acceso>
[ON <lista_de_objetos>]
TO <lista_de_usuarios> ;

2.3.2 CONTROL DE INTEGRIDAD

Este control está asociado al concepto de Transacción. Existen dos sentencias principales:

- **COMMIT**: Los cambios que se puedan estar realizando sobre la base de datos se hacen fijos únicamente al completar la transacción (COMMIT automático) o al hacer un COMMIT explícito.
- **ROLLBACK**: Elimina todos los cambios que se hayan podido producir en la base de datos desde la ejecución de la última instrucción COMMIT. Si se produce un error de programa o un fallo hardware el sistema realiza un ROLLBACK automáticamente.

3. ESTANDARES DE CONECTIVIDAD: ODBC Y JDBC

Los programas de aplicación son programas que se usan para interaccionar con la base de datos. Como ya se comentó, estos programas se escriben usualmente en un lenguaje anfitrión, tal como Cobol, C, C++ o Java.

Para acceder a la base de datos relacionales, las instrucciones del LMD del SQL necesitan ser ejecutadas desde el lenguaje anfitrión. Hay dos maneras de hacerlo:

- SQL incorporado: Extendiendo la sintaxis del lenguaje anfitrión para incorporar las llamadas del LMD dentro del programa del lenguaje anfitrión. Usualmente, un carácter especial o una sentencia concreta precede a las llamadas del LMD y un precompilador LMD, convierte las instrucciones LMD en llamadas normales a procedimientos del lenguaje anfitrión.
- SQL dinámico: Proporcionando una interfaz de programas de aplicación, API (Application Program Interface), que se deben usar para enviar tanto instrucciones LMD, como LDD, a la base de datos, y recuperar los resultados. Existen dos estándares:
  - El estándar de conectividad abierta de base de datos (ODBC, Open Data Base Connectivity) definido por Microsoft para el uso con el lenguaje C, usado comúnmente.
  - El estándar de conectividad de Java con base de datos (JDBC, Java Data Base Connectivity) que proporciona las características correspondientes para el lenguaje Java.

En el resto del apartado, vamos a examinar las dos normas de conexión a base de datos, ODBC y JDBC, utilizando el lenguaje SQL.

Para comprender estas normas es necesario comprender el concepto de sesión SQL. El usuario o aplicación se conecta a un servidor de base de datos SQL, estableciendo una sesión. Así todas las actividades del usuario o aplicación están en el contexto de una sesión SQL. Además de las ordenes normales de SQL (LMD y LDD), una sesión también puede contener órdenes para comprometer el trabajo realizado hasta ese momento en la sesión (COMMIT) o para echarlo atrás (ROLLBACK).

Las normas ODBC y JDBC, se desarrollaron para hacer de interfaz entre clientes y servidores. Cualquier cliente que utilice interfaces ODBC o JDBC puede conectarse a cualquier servidor de base de datos que proporcione esta interfaz.

3.1 ODBC

¿Qué es ODBC?

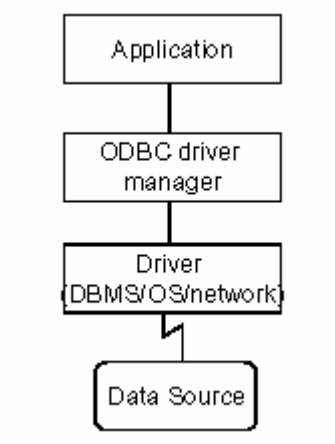
- ODBC son las siglas de Open Database Connectivity.
- Es un interface estándar de programas de aplicación (API) para acceso a base de datos.
- Impulsado por SQL Access Group, principalmente por Microsoft, desde 1992.
- Usando sentencias ODBC en un programa, se puede acceder a las tablas de diferentes bases de datos, tales como: Access, dBase, DB2, etc.

- Permite a los programas utilizar sentencias SQL que acceden a las bases de datos, sin tener que conocer los interfaces propietarios de dichas bases de datos.
- ODBC maneja la sentencia SQL requerida y la convierte en una petición a la base de datos.
- No soporta el COMMIT en dos fases, para coordinar el acceso simultaneo a varias bases de datos

ODBC presenta una arquitectura de cuatro niveles:

- La aplicación propiamente dicha.
- ODBC driver manager: Módulo separado por cada base de datos a la que se quiere acceder. A este módulo es al que se conecta dinámicamente la aplicación.
- Driver DBMS/OS/Network: es un controlador que hace transparente el gestor de base de datos, el sistema operativo y los protocolos de red.
- El propio servidor de datos o fuente de datos.

Esta arquitectura, es la que muestra la figura:



Las aplicaciones como las interfaces gráficas de usuario, los paquetes estadísticos y las hojas de cálculo pueden usar la misma API ODBC, para conectarse a cualquier servidor de base de datos compatible con ODBC.

Cada sistema de base de datos que sea compatible con ODBC proporciona una biblioteca que se debe enlazar con el programa cliente. Cuando este programa cliente realiza una llamada a la API ODBC, el código de la biblioteca se comunica con el servidor de base de datos para realizar la acción solicitada y obtener los resultados.

ODBC se basa en las normas SQL de Interface de nivel de llamada, CLI (Call-Level Interface) desarrolladas por el consorcio industrial X/Open y el grupo SQL Access, pero tienen varias extensiones. La API ODBC define una CLI, una definición de sintaxis SQL y reglas sobre las secuencias admisibles de llamadas CLI. La norma también define los niveles de conformidad para CLI y la sintaxis SQL:

- El nivel central de la CLI tiene comandos para conectarse con bases de datos, para preparar y ejecutar sentencias SQL, para devolver resultados o valores de estado y para administrar transacciones.
- El nivel uno, siguiente nivel de conformidad, exige el soporte de la recuperación de información de los catálogos de los SGBD, como la información sobre las relaciones existentes y los tipos de sus atributos, y otras características que superan la CLI del nivel central.
- El nivel dos exige más características, como la capacidad de enviar y recuperar arrays de valores de parámetros y de recuperar información de catálogo más detallada.

3.2 JDBC

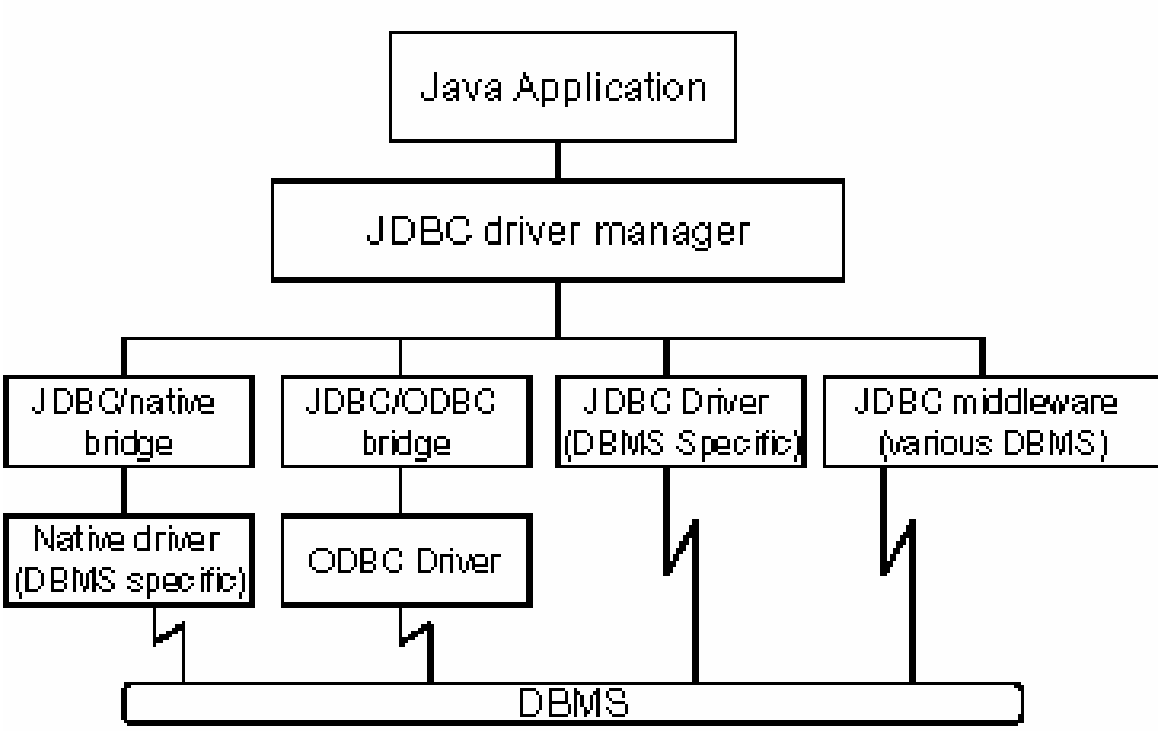
¿Qué es JDBC?

- JDBC son las siglas de Java Database Connectivity.
- Es un Java API, para conectar programas escritos en Java a datos almacenados en SGBDR.
- Consiste en un conjunto de clases e interfaces escritos en el lenguaje de programación Java.
- Suministra un API estándar para los programadores, haciendo posible desarrollar aplicaciones con acceso a bases de datos usando un “puro” Java API.
- Este estándar es definido por Sun Microsystems, y permitiendo a los diversos suministradores de base de datos, implementar y extender dicho estándar con sus propios JDBC drivers.
- JDBC establece una conexión con la base de datos, envía las sentencias SQL y procesa los resultados.
- Con un pequeño programa “puente” se puede utilizar el interface JDBC, para acceder a las bases de datos a través de ODBC.

Pasos que hay que realizar en un programa Java utilizando el API JDBC:

- Importación de paquetes: Estos paquetes contienen el propio JDBC y los drivers para una determinada base de datos.
- Registrar los drives de JDBC: DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver()); En este caso se registra un driver para acceder a Oracle.
- Abrir una conexión a la base de datos: Connection conn = DriverManager.getConnection ("jdbc:oracle:thin:@aardvark:1526:teach", user, password); Se indica el tipo de driver, nombre de host, puerto, nombre de la base de datos, usuario y password, es decir, todo lo necesario para localizar la base de datos y poder acceder a ella.
- Crear un objeto de tipo Statement: PreparedStatement pstmt = conn.prepareStatement (x); En este caso x es la sentencia SQL que se quiere ejecutar.
- Ejecutar la query o sentencia SQL, y devolver el resultado en un objeto de tipo Result Set: ResultSet rset = pstmt.executeQuery ();
- Procesar el Result Set que nos ha devuelto la base de datos.
- Cerrar los objetos creados: Resul Set y Statement.
- Cerrar la conexión.

En la siguiente figura, se muestra las cuatro arquitecturas JDBC que existen actualmente:



Se reflejan, los cuatro tipos de driver con los que puede trabajar JDBC. Son:

- Driver tipo 1: JDBC/ODBC bridge. Acceso a base de datos a través del API ODBC.
- Driver tipo 2: JDBC Driver (DBMS específico). Acceso directo a una base de datos concreta.
- Driver tipo 3: JDBC/native brigde. Acceso a base de datos a través de un driver java nativo, que está arrancado en la parte del Servidor.
- Driver tipo 4: JDBC middleware. Acceso directo a varias bases de datos. Soporte de COMMIT en dos fases para coordinar las actualizaciones en las diversas bases de datos.

4. CONCLUSIÓN

En este tema hemos estudiado los lenguajes de interrogación de bases de datos y hemos visto tres lenguajes formales. El primer lenguaje de consulta, el álgebra relacional, se ha estudiado en detalle. El álgebra relacional forma la base del lenguaje de consulta SQL, ampliamente usado. A continuación se proporcionado una visión general de otros dos lenguajes formales: el cálculo relacional de tuplas y el cálculo relacional de dominios, que son lenguajes declarativos de consulta basados en la lógica matemática. El cálculo relacional de dominios es la base del lenguaje de consulta QBE.

Los lenguajes formales proporcionan una notación concisa para la representación de las consultas. Sin embargo, los sistemas de bases de datos comerciales necesitan un lenguaje de consulta cómodo para el usuario. Hemos estudiado el lenguaje comercial que actualmente tiene mayor influencia, SQL. SQL es una combinación de álgebra relacional y construcciones de cálculo relacional.

Finalmente se ha tratado los dos estándares de conectividad asociados a SQL: ODBC y JDBC. El primero para lenguaje C y el segundo para lenguaje Java.

5. BIBLIOGRAFÍA

- De Miguel y Piattini (1999): Fundamentos y modelos de bases de datos. Ra-Ma.
- A. Silberschatz, H. F. Korth y S. Sudarshan (1998): Fundamentos de Bases de Datos". Mc Graw-Hill.
- C.J. Date (1993): Introducción a los Sistemas de Bases de Datos Volumen I, Quinta Edición. Addison-Wesley Iberoamericana.
- Rick Cattell (1997): Object Database Standard ODMG 2.0. Object Database Management Group.

6. ESQUEMA – RESUMEN

El álgebra relacional es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

Operaciones fundamentales del álgebra relacional:

- **Unarias:** Porque operan con una sola relación o tabla. Son:
  - Selección: Si P es un predicado compuesto por operadores lógicos, aritméticos y de comparación y sus operandos son los valores de los atributos de una relación R, entonces la selección $\sigma_P(R)$ es el conjunto de tuplas de la relación R que hacen verdadera la condición establecida por el predicado P.
  - Proyección: Si x es un subconjunto de atributos de la relación R, entonces la proyección $\pi_x(R)$ es la relación formada por las columnas de R correspondientes a los atributos x.
  - Renombramiento: Dada una expresión E del álgebra relacional, la expresión $\rho_x(E)$, devuelve el resultado de la expresión E con nombre x
- **Binarias:** Porque operan con dos relaciones. Son:
  - Unión: La unión de dos relaciones R y S (R U S) es el conjunto formado por todas las tuplas de R más todas las tuplas de S. Este operador sólo se puede aplicar a relaciones del mismo grado y con los mismos atributos.
  - Diferencia de conjuntos: La diferencia de dos relaciones R y S (R - S) es el conjunto formado por todas las tuplas de R que no están en S.
  - Producto cartesiano: La definición formal dice: El producto cartesiano de dos relaciones R y S, de grados *m* y *n* respectivamente, se denota R x S y es el conjunto formado por todas las posibles tuplas de *m* + *n* atributos en las que los *m* primeros atributos son de R y los *n* restantes pertenecen a S.

Otras operaciones derivadas de las fundamentales:

- La operación intersección de conjuntos denotada por ∩, permite buscar la tuplas que estén al tiempo en las dos relaciones sobre las que actúa.
- La operación unión natural forma un producto cartesiano de sus dos argumentos, realiza una selección forzando la igualdad de los atributos que aparecen en ambas relaciones y, finalmente elimina los atributos duplicados.
- La operación asignación, denotada por ←, actúa de manera parecida a la asignación de los lenguajes de programación.

Las consultas se expresan en el cálculo relacional de tuplas como: $\{t \mid P(t)\}$, es decir, son el conjunto de todas las tuplas tales que el predicado P es cierto para *t*. Se utiliza la notación *t*[A] para denotar el valor de la tupla *t* en el atributo A y $t \in R$ para denotar que la tupla *t* está en la relación R.

Hay una segunda forma de cálculo relacional denominada cálculo relacional de dominios. Esta forma utiliza variables de dominio que toman sus valores del dominio de un atributo, en vez de tomarlos de una tupla completa. El cálculo relacional de dominios, sin embargo, se halla estrechamente relacionado con el cálculo relacional de tuplas.

El lenguaje SQL es un lenguaje de alto nivel para dialogar con los SGBD-R. Como todo lenguaje de un SGBD, esta formado por tres componentes claramente diferenciados:

- Lenguaje de definición de datos: CREATE, ALTER Y DROP.
- Lenguaje de manipulación de datos: INSERT, UPDATE, DELETE Y SELECT.
- Lenguaje de control de datos: GRANT, REVOKE, COMMIT Y ROLLBACK.

Existen dos estándares de conectividad para SQL:

- El estándar de conectividad abierta de base de datos (ODBC, Open Data Base Connectivity) definido por Microsoft para el uso con el lenguaje C, usado comúnmente.
- El estándar de conectividad de Java con base de datos (JDBC, Java Data Base Connectivity) que proporciona las características correspondientes para el lenguaje Java.

¿Qué es ODBC?

- ODBC son las siglas de Open Database Connectivity.
- Es un interface estándar de programas de aplicación (API) para acceso a base de datos.
- Impulsado por SQL Access Group, principalmente por Microsoft, desde 1992.
- Usando sentencias ODBC en un programa, se puede acceder a las tablas de diferentes bases de datos, tales como: Access, dBase, DB2, etc.
- Permite a los programas utilizar sentencias SQL que acceden a las bases de datos, sin tener que conocer los interfaces propietarios de dichas bases de datos.
- ODBC maneja la sentencia SQL requerida y la convierte en una petición a la base de datos.
- No soporta el COMMIT en dos fases, para coordinar el acceso simultaneo a varias bases de datos

¿Qué es JDBC?

- JDBC son las siglas de Java Database Connectivity.
- Es un Java API, para conectar programas escritos en Java a datos almacenados en SGBDR.
- Consiste en un conjunto de clases e interfaces escritos en el lenguaje de programación Java.
- Suministra un API estándar para los programadores, haciendo posible desarrollar aplicaciones con acceso a bases de datos usando un “puro” Java API.
- Este estándar es definido por Sun Microsystems, y permitiendo a los diversos suministradores de base de datos, implementar y extender dicho estándar con sus propios JDBC drivers.
- JDBC establece una conexión con la base de datos, envía las sentencias SQL y procesa los resultados.
- Con un pequeño programa “puente” se puede utilizar el interface JDBC, para acceder a las bases de datos a través de ODBC.